# Software Quality Management

**Dr. Stefan Wagner**
Technische Universität München

Garching
18 June 2010

## Last QOT: Why is software reliability a random process?

Software reliability is mostly considered a random process, because the inputs chosen by
end users in practice are randomly distributed.

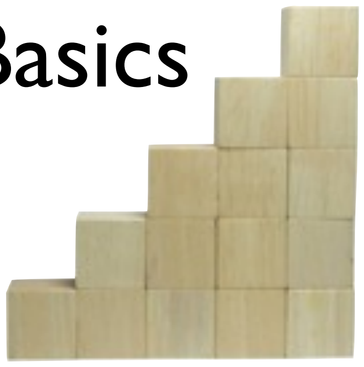
New QOT: "Do CMMI level 5 companies produce software with higher quality than CMMI level 1 companies?"

## Reliability models

## Quality measures

Review of last week's lecture.

Basics



Product

Quality

Metrics and

Measurement

Process

Quality

Quality

Management

Certifi-
cation

We are in the part "Metrics and Measurement".

## Reliability models

## Quality measures

## Visualisation

This lecture finishes quality measures and covers visualisation of quality measures.

## Halstead's software science

Program = Operations + Operands

|  |  |  |
|---|---|---|
| Base measures: | n | # different operators |
|  | m | # different operands |
|  | N | # operators |
|  | M | # operands |

|  |  |  |
|---|---|---|
| Derived measures: | Length | $L = N + M$ |
|  | Volume | $V = L \times \log_2 (n + m)$ |
|  | Complexity | $C = (n / 2) \times (M / m)$ |
|  | Test effort | $E = V \times C$ |

Halstead proposed these measures with the aim to measure the complexity for maintenance and comprehensionof source code. He has never empirically validated the measures.

There are some reports of useful usage of these measures, but this could be random success.

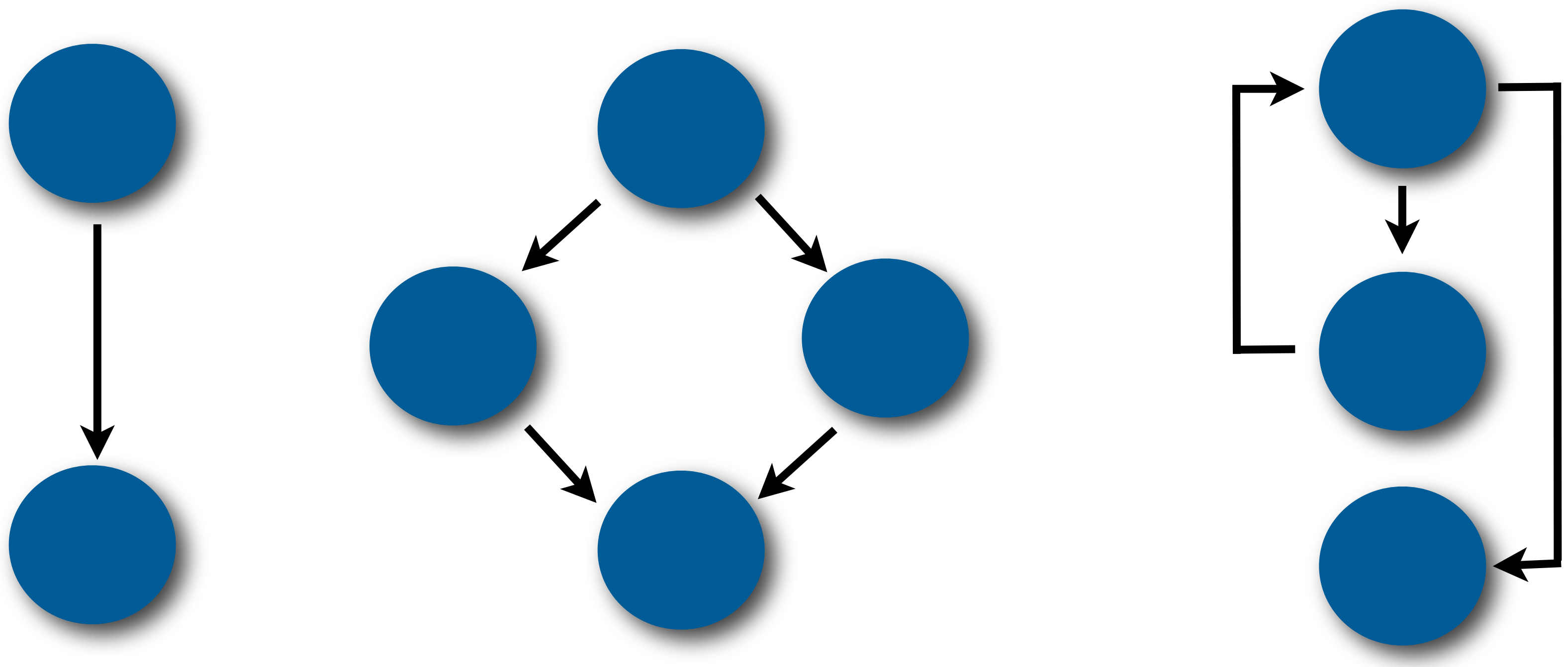
## McCabe's cyclomatic complexity

$v \in V$: statements

$e \in E$: control flow dependency

$V(g) = |E| - |V| + 2p$

p: connected components



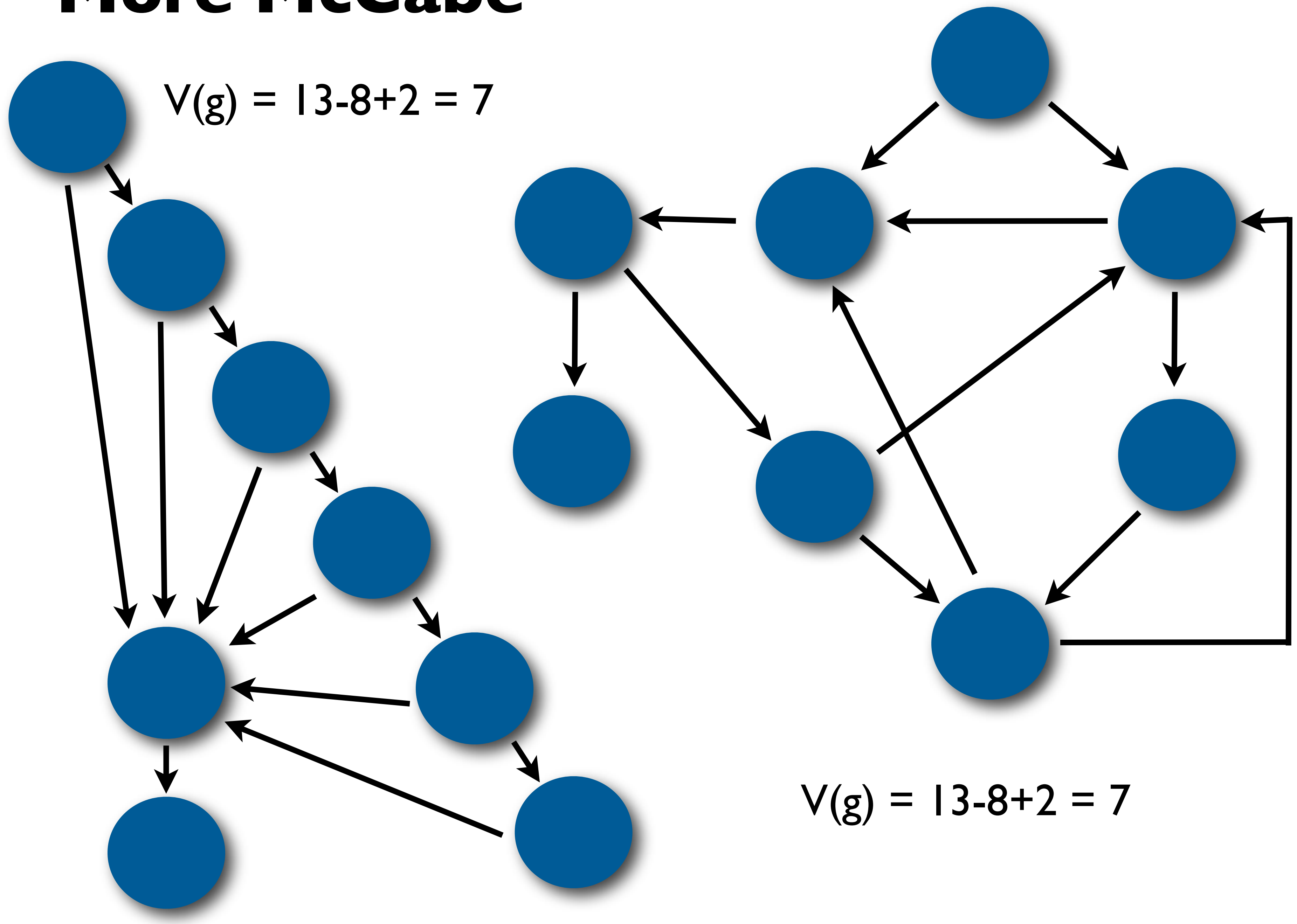
$V(g) = 1-2+2 = 1$ $V(g) = 4-4+2 = 2$ $V(g) = 3-3+2 = 2$

This is probably the most known complexity measure for source code. The aim again was to measure the complexity of the code for comprehending and changing it.

It operates on the control flow graph and hence analyses the structure of the code.

The program is the graph G = (E, V)

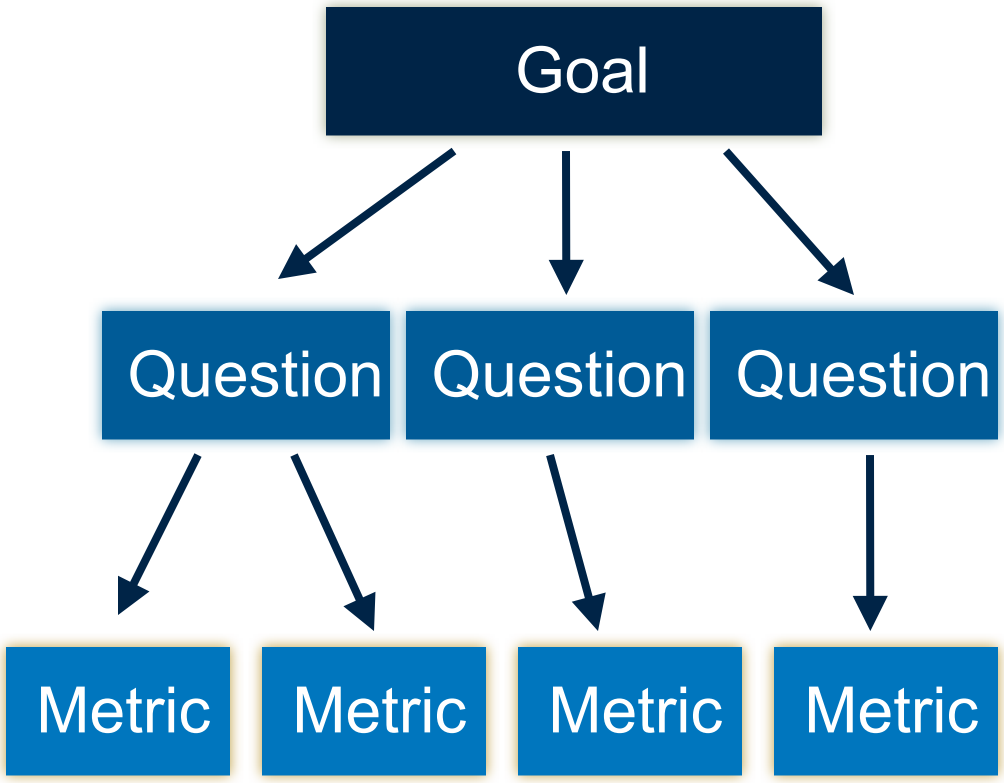
## More McCabe



This example and various empirical investigations show that it is easy to forge McCabe's complexity
measure.

Nevertheless, it measures the number of linearly independent paths in the code. It is questionable to
what extent this influences comprehension and modification, but it is a useful measure for testability.
The more paths, the more difficult is it to the test the function.

## GQM



We have seen a variety of quality measures in this lecture. Many of them can be collected

automatically. This may lead to unsystematic collections of quality data, especially of measures

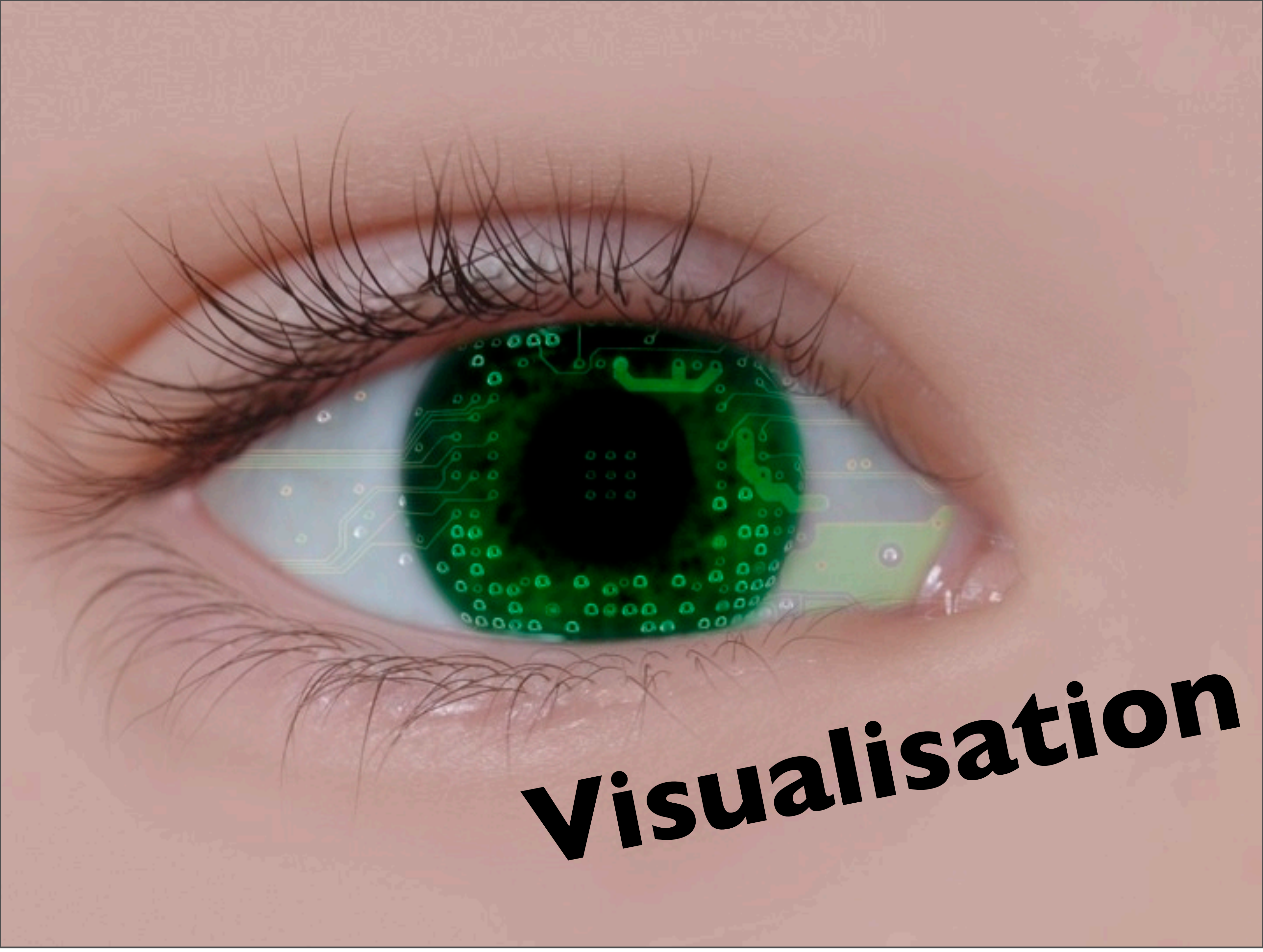
that are easy to measure.

What is interesting is usually hard to measure, what is easy to measure is usually not interesting!

## Reliability models

## Quality measures

## Visualisation



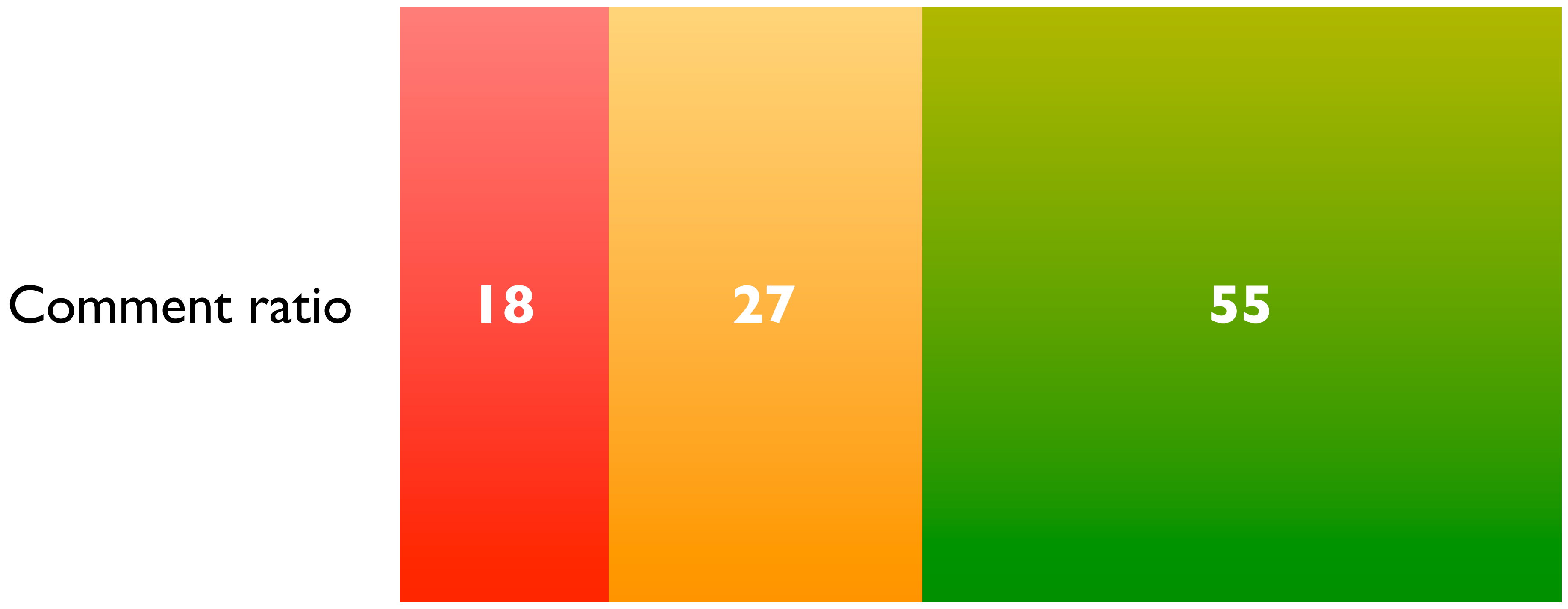
## Visualisation

Even if we employ a structured, goal-driven measurement program, the results can be huge.

For large systems, it is easy to get lost in the amount of data.

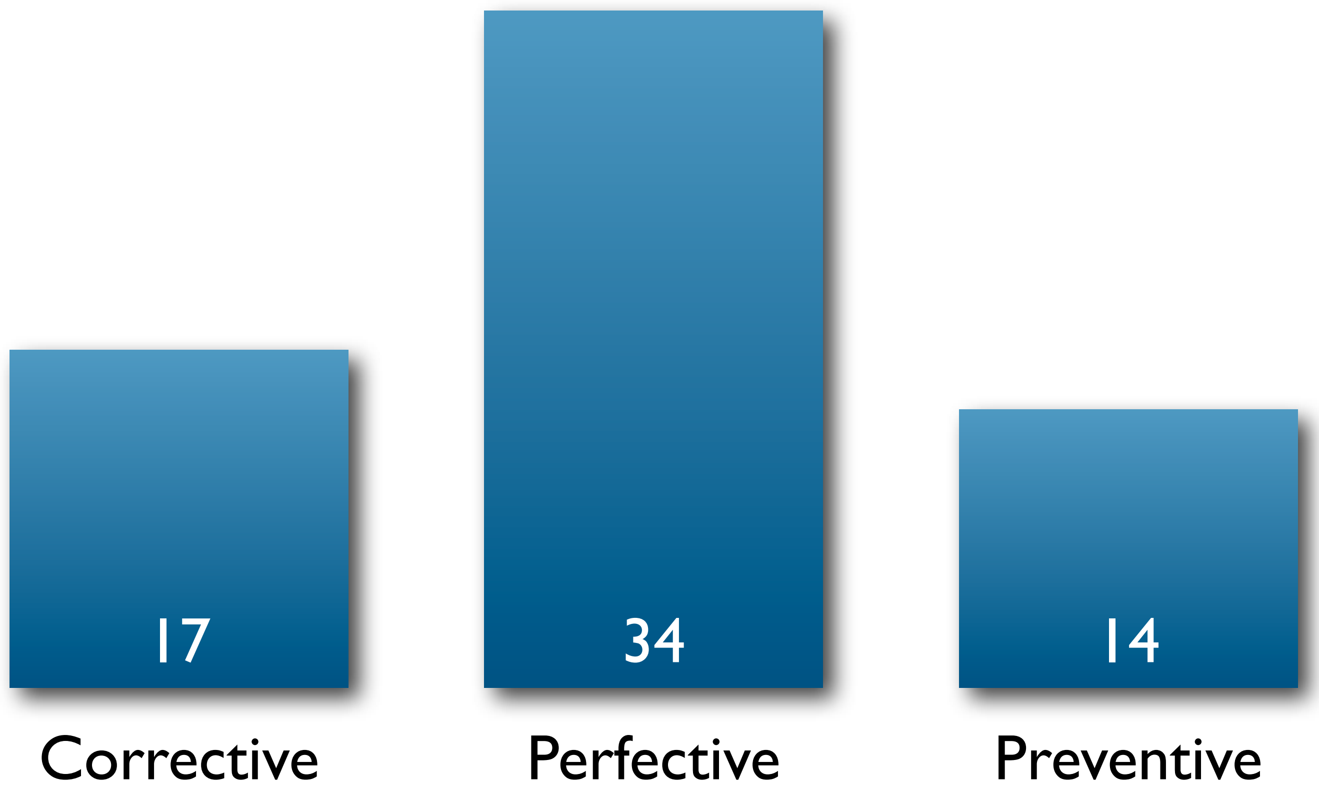
Humans are visual beings and visualisation helps to get an overview and to find problems

## Traffic lights



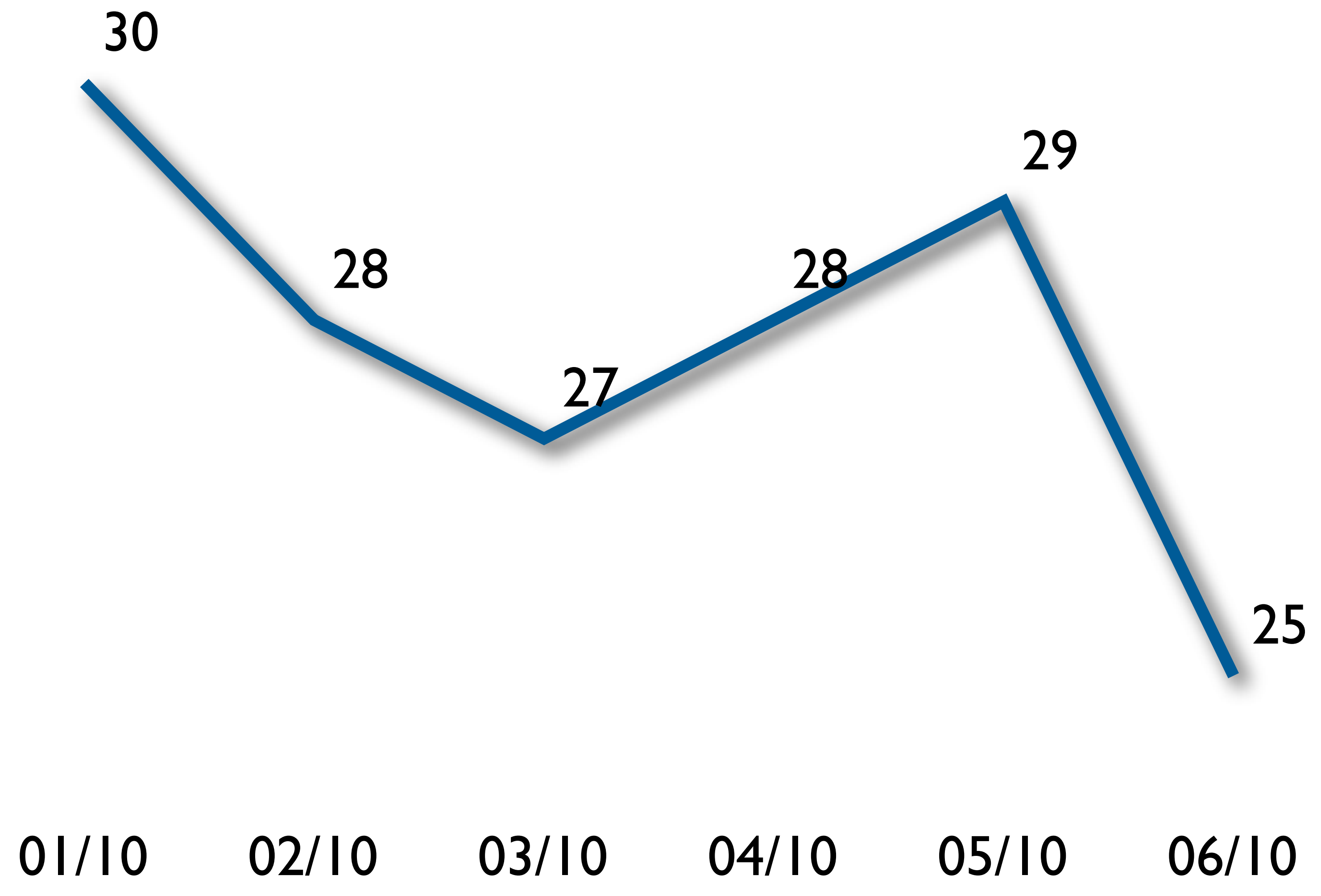
A visualisation that often can be a crude representation of the data, but which is useful for a
very quick overview are traffic lights. Formally speaking, it is usally a rescaling aggregation. Some
complicated data in interval or ratio is aggregated into an ordinal scale of red, yellow, and green.
Green usually represents the values that are "good", i.e., on an accepted level, yellow are problematic
values, and red are "bad" values.
The example shows the percentage of classes in a software system with comment ratios on the
red, yellow, and green level. Red can be defined as a comment ratio between 0 and 0.1, yellow as
0.1 to 0.2, and red above 0.2.
Another use of traffic lights with only red and green is to show the percentage of unit tests that
passed and failed.

## Bar chart



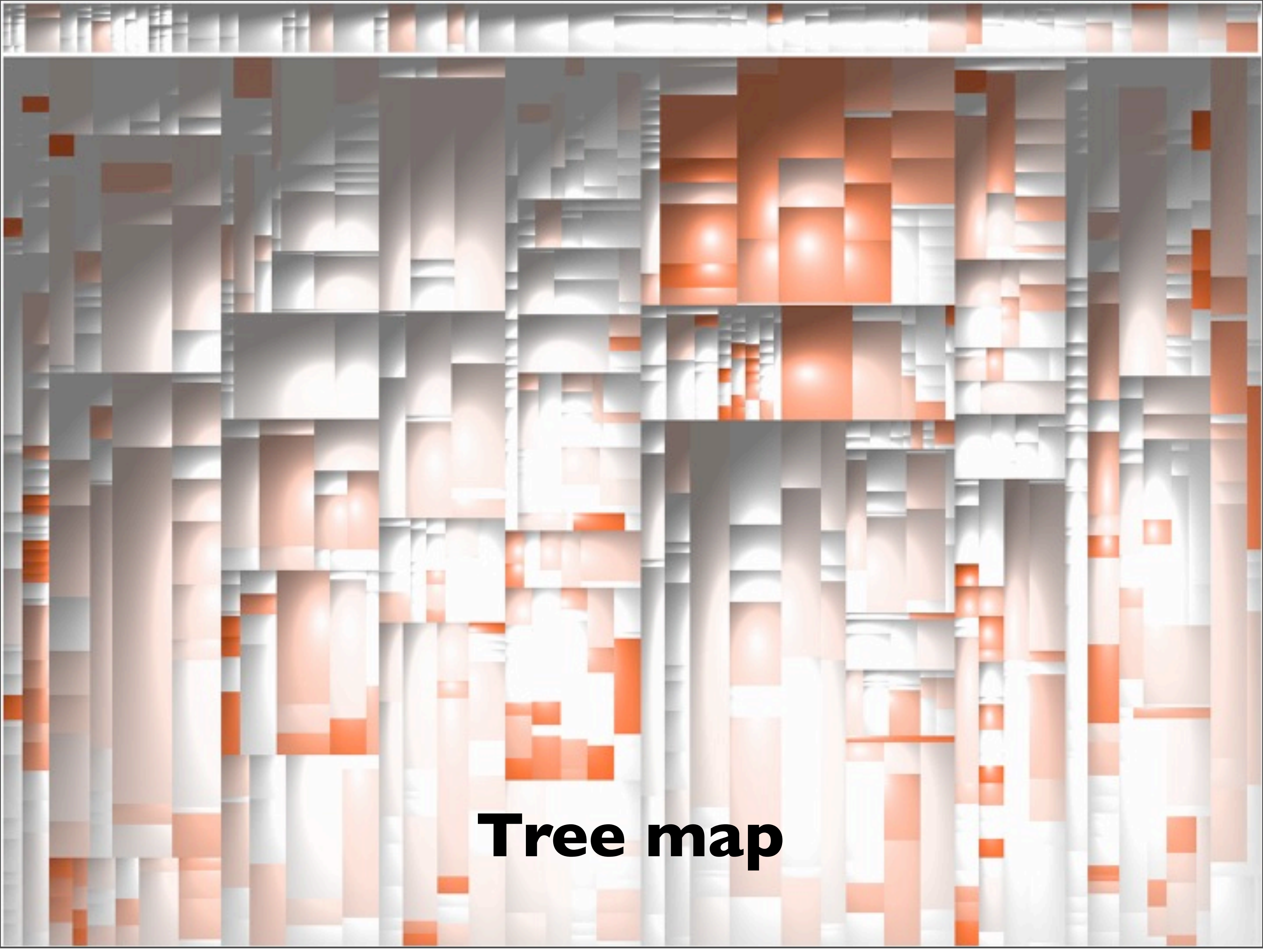
A bar chart is a useful visualisation for comparing different values. The example compares the
number of change requests for each type of change request. This shows us whether we mostly
fix bugs (corrective) or if we are able to build new features (perfective) and restructure and
improve the internals of the system (preventive).

## Trend chart



Trend charts are visualisations that help to track measures over time. In many cases, I am
not only interested in the current state of my system, but also in the change over time.
This is especially useful for measures that indicate problems, such as clone coverage or
warnings from bug pattern tools. Often it is not possible and also not adivsable to make
immediate huge changes to the system to remove all these problems. Nevertheless, it
makes sense to observe these measures over time and to make sure that at least they
do not increase or even better that they decrease.

The example shown is the percentage of clone coverage of a system over six month.



## Tree map

Tree maps are mostly used for finding hot spots in the system. The intensity of the colour indicates

the intensity of the measure. The example here is again clone coverage. As this is a probability, it

is in the range 0 to 1. The visualisation is than white for 0 and completely red for 1. Any shades

between show different levels of clone coverage. The tree map has the additional dimension of the

squares it shows. These squares can show another measure, usually the size of classes or components.

## Reliability models

## Quality measures

## Visualisation

Basics

Product Quality

Metrics and Measurement

Process Quality

Quality Management

Certifi-cation

We now move into the part "Process Quality".

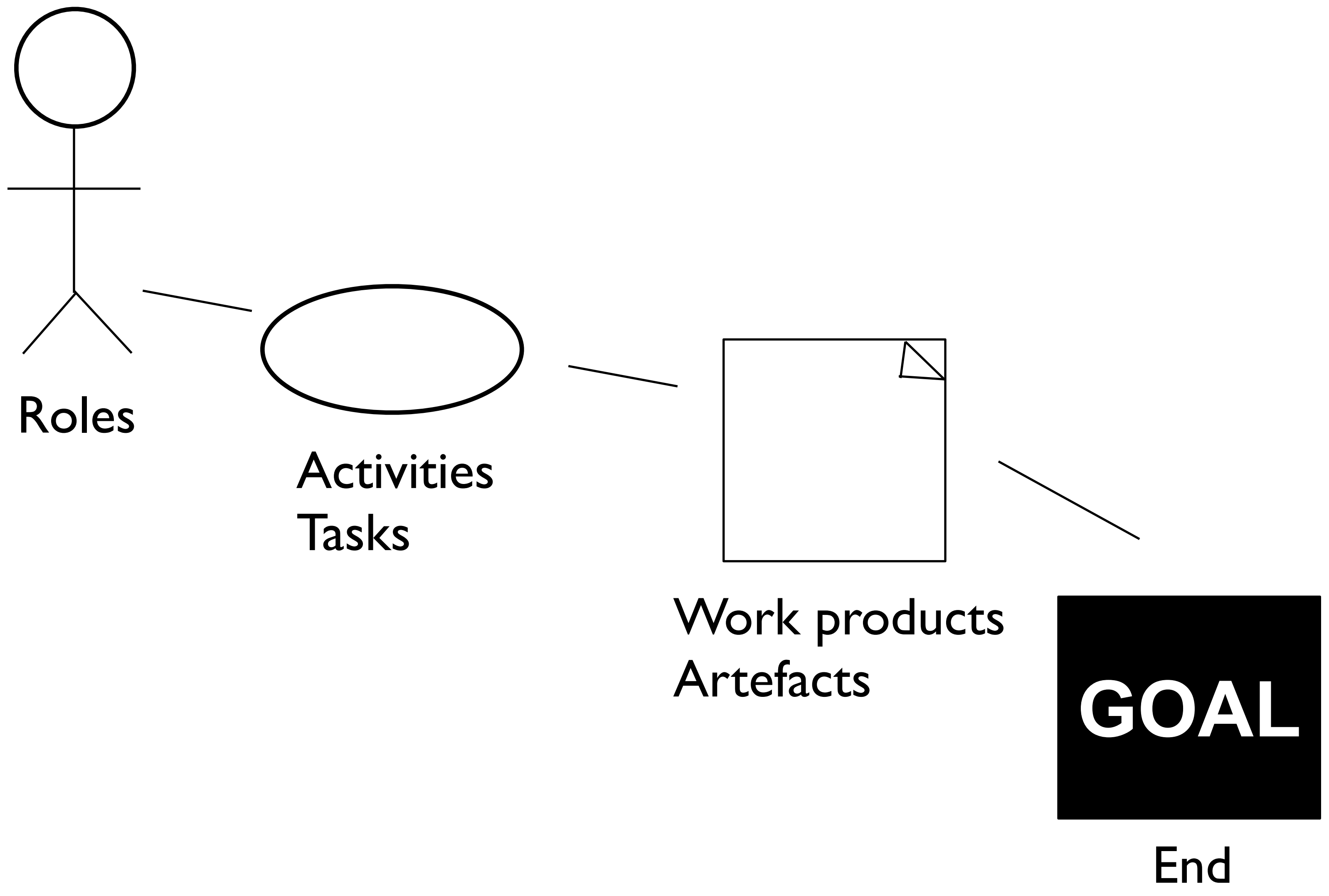
**vs.**

What is the difference?
What is more important?

We have already discussed the difference between product and process quality.

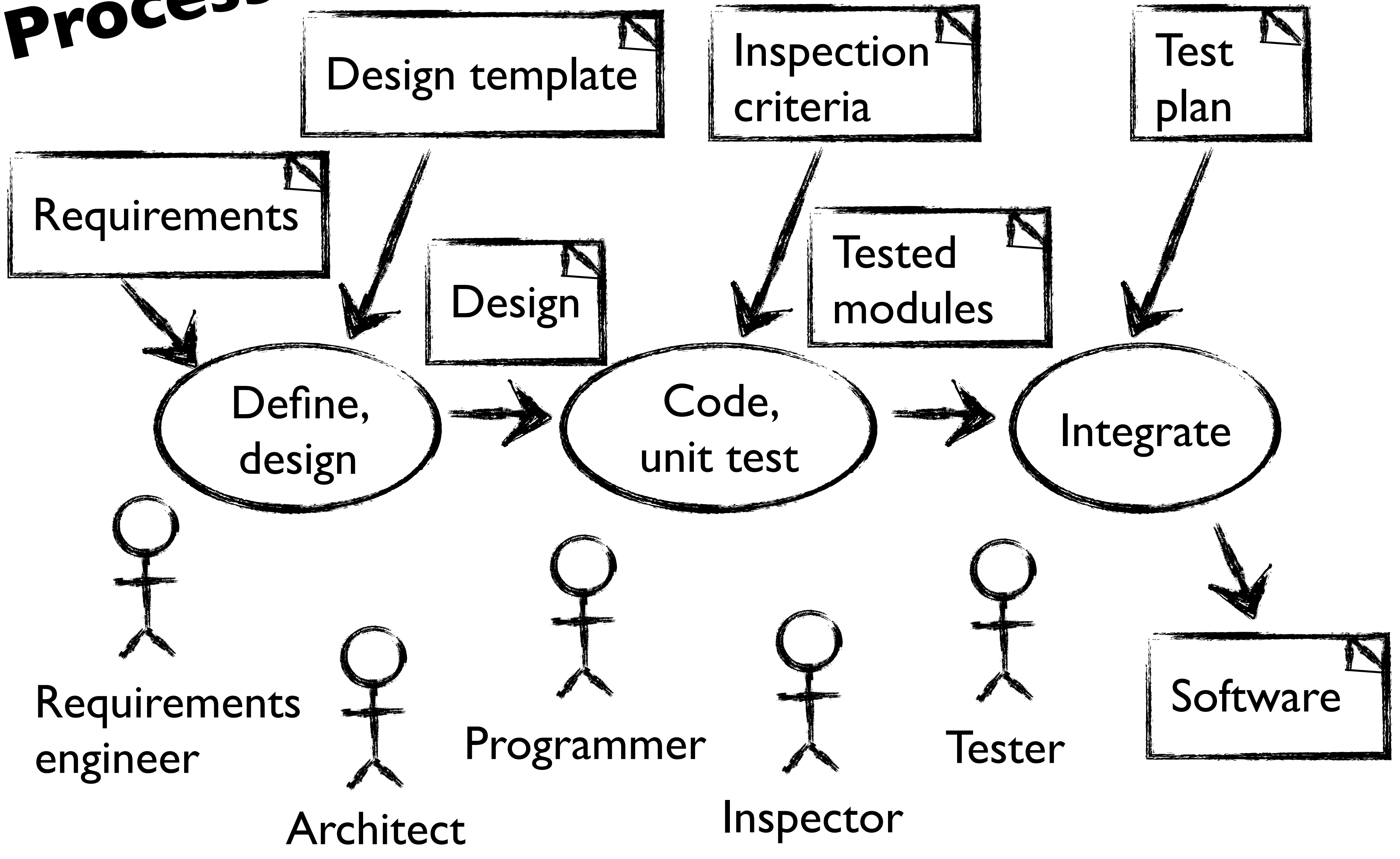
Process quality is one important prerequisite for product quality.

## What is a process?



A process consists in its core of activities (or tasks) that produce work products (or artefacts). This production of work products by activities is done by people that are categorised in roles. The whole process always has to have the aim to reach a certain goal; the end for which the process is the means.

# Process sketch



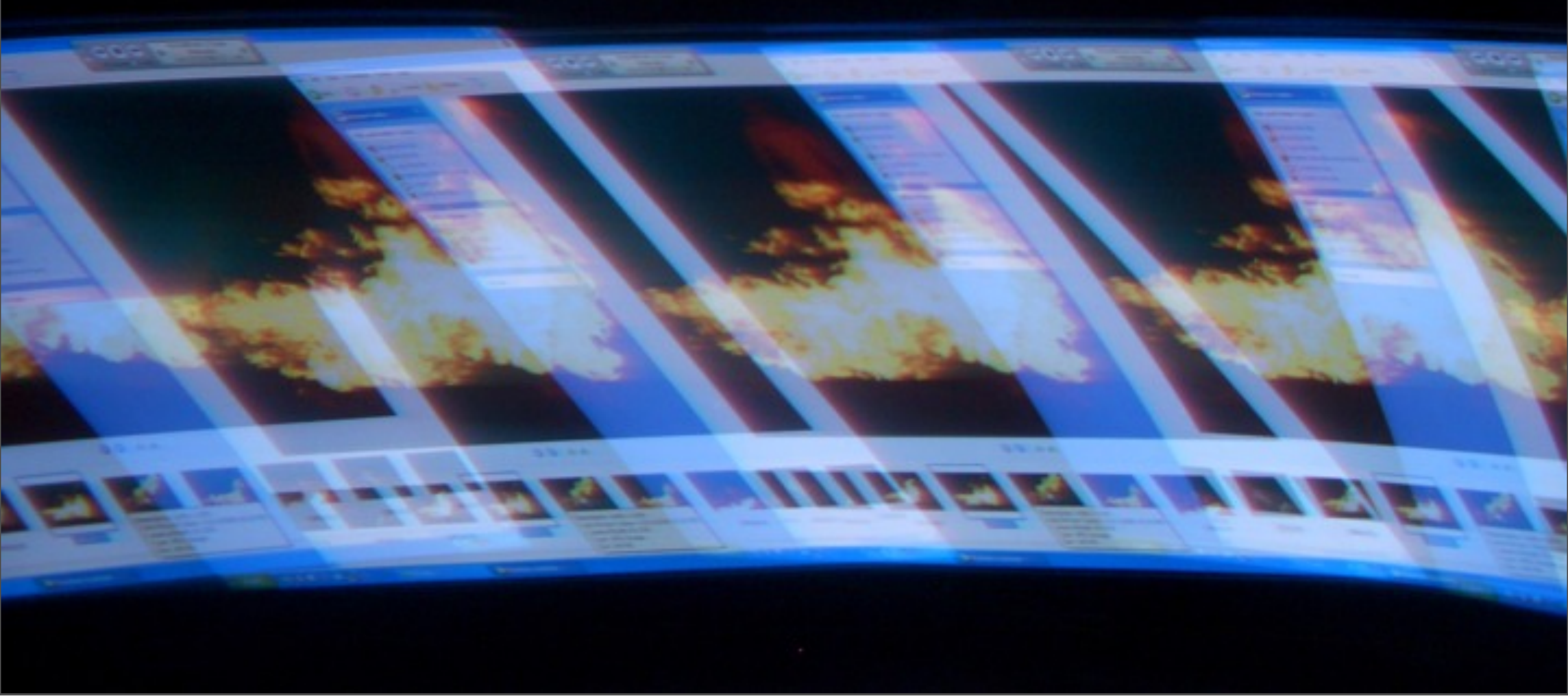
This process sketch is based on Pfleeger, Atlee (2010). It shows a simplified example how a software development process can look like.

**Post mortem analysis**

**ISO 9000**

**CMMI/SPICE**

These are the three topics in process quality that we will cover today.

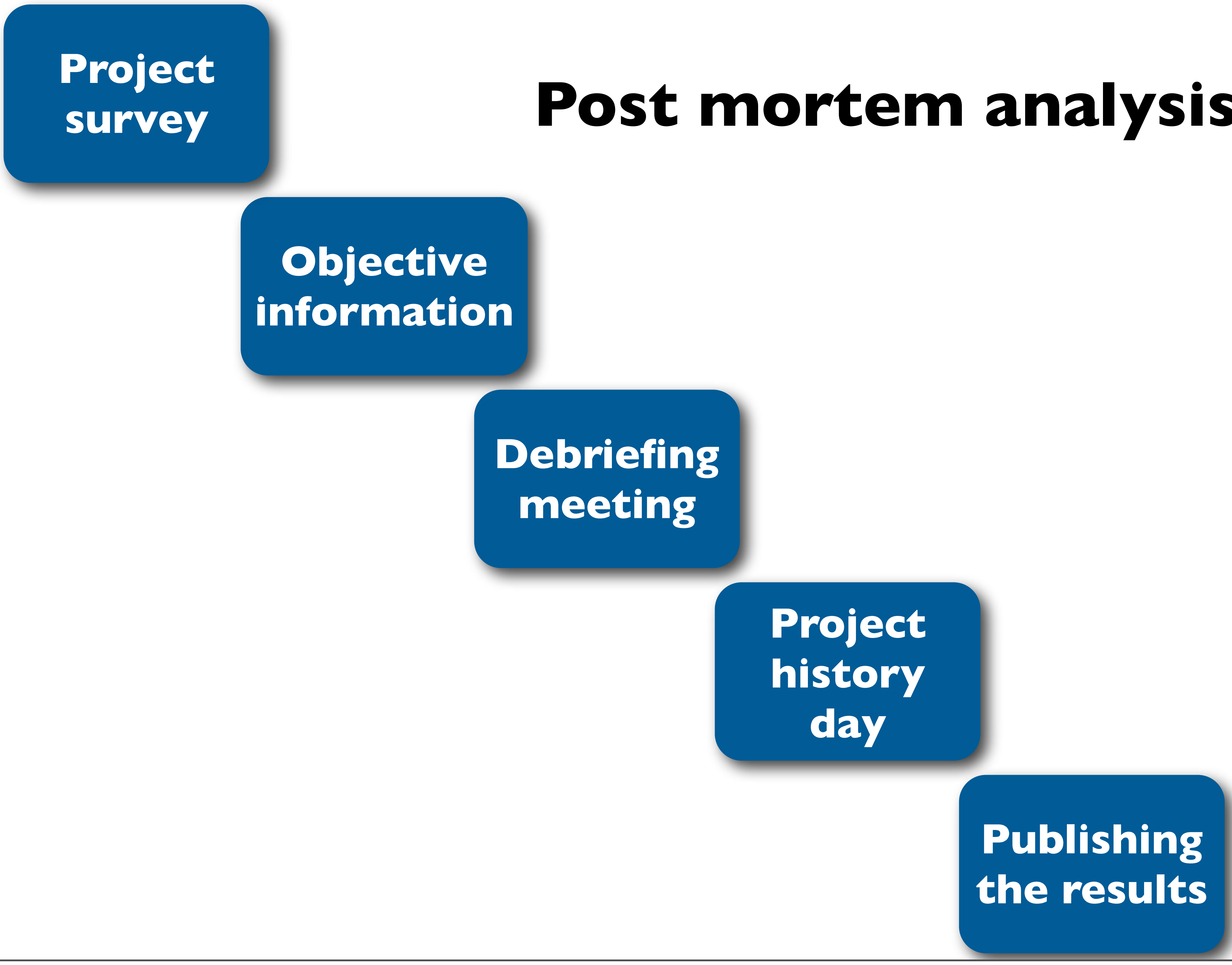


## Those who cannot remember the past are condemned to repeat it.

–George Santayana

This is not only true in politics, but also in software engineering. In essence: Learn from the mistakes you have made in the past to avoid them in the future.

## Post mortem analysis



Such an analysis is not only useful after the project is completely finished, but also after larger iterations.
We learn a lot from our successes, but even more from our failures.

Postmortem process from Collier, DeMarco, and Fearey (1996):
1. Design and promulage a project survey to collect data without compromising confidentiality.
2. Collect objective project information, such as resource costs, boundary conditions, schedule predictability, and fault counts.
3. Conduct a debriefing meeting to collect information the survey missed.
4. Conduct a project history day with a subset of project participants, to review project events and data and to discover key insights.
5. Publish the results by focusing on lessons learnt.

## Pair work

- Develop a questionnaire for the post mortem survey of the last project(s) that you took part in!
- Discuss with your neighbour.
- Write on each post-it one question.
- 10 minutes
- When you finish, put your question on the whiteboard.
- Arrange them so that similar question are close.

Proposed categories from Collier, DeMarco, and Fearey (1996):
Support and goals
Expectations and communications
Issues resolution
Information access
Product specifications
Engineering practices
The big picture
Demographics

Did you plan enough time for testing?
What was the main risk you had?
How to communicate with your partners?
How would you rate how the project went in school grades?
How much time did you need for communication?
Likert
I planned time for testing
totally agree

The best information is usually gathered by asking about personal experiences. For example:
- Did you learn something in the project?
- Did you enjoy working in the project?

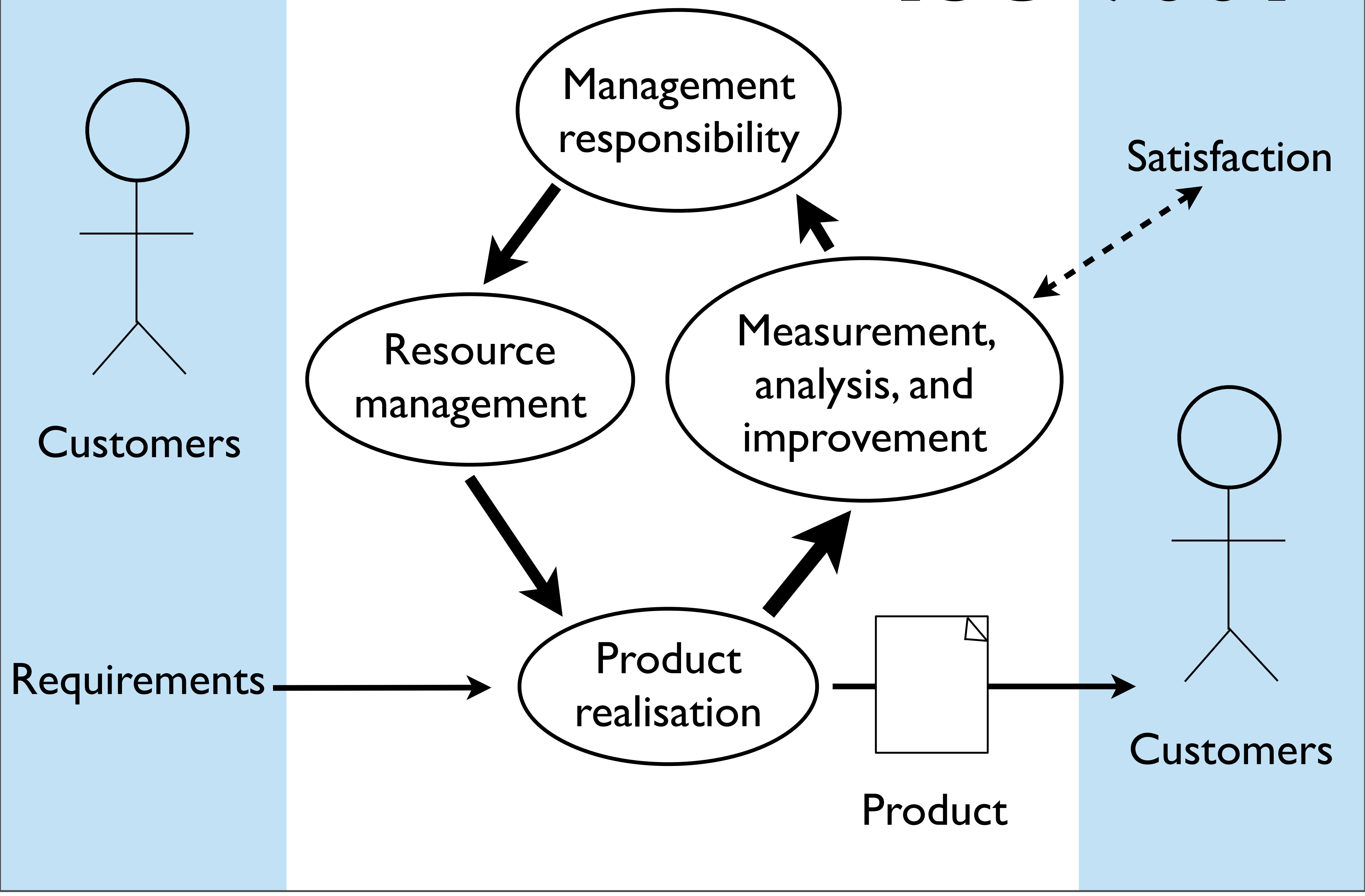
If you want quantitative information, which is also easier to analyse, you need to give a scale for the answers.
The most common scale for that is the Likert scale that expresses agreement. For example:
I planned enough time for testing
Totally agree – partially agree – neither agree nor disagree – partially disagree – totally disagree

## Post mortem analysis

ISO 9000

CMMI/SPICE

## ISO 9001



This International Standard specifies requirements for a quality management system where an organization
a) needs to demonstrate its ability to consistently provide product that meets customer and applicable regulatory requirements, and
b) aims to enhance customer satisfaction through the effective application of the system, including processes for continual improvement of the system and the assurance of conformity to customer and applicable regulatory requirements.

## Quality management system

- Identification and application of necessary processes
- Sequence and interactions of processes
- Criteria and methods to make processes effective
- Resources and information to monitor processes
- Montoring, measurment, analysis
- Continual improvement
- Quality manual
- Control of documents and records

ISO 9001 is document-intensive. Certifications check whether you produce all necessary documents.

## Management responsibility

- Management commitment
- Customer focus
- Quality policy
- Planning (quality objectives, QMS)
- Responsibility, authority, and communication
- Management review

QMS stands for quality management system.

## Ressource management

- Provision of resources
- Human resources
- Infrastructure
- Work environment

## Product realisation

- Planning of product realisation
- Customer-related processes
- Design and development
- Purchasing
- Production and service provision
- Control of monitoring and measuring devices

## Measurement, analysis, improvement

- Monitoring and measurement
- Control of nonconforming product
- Analysis of data
- Improvement

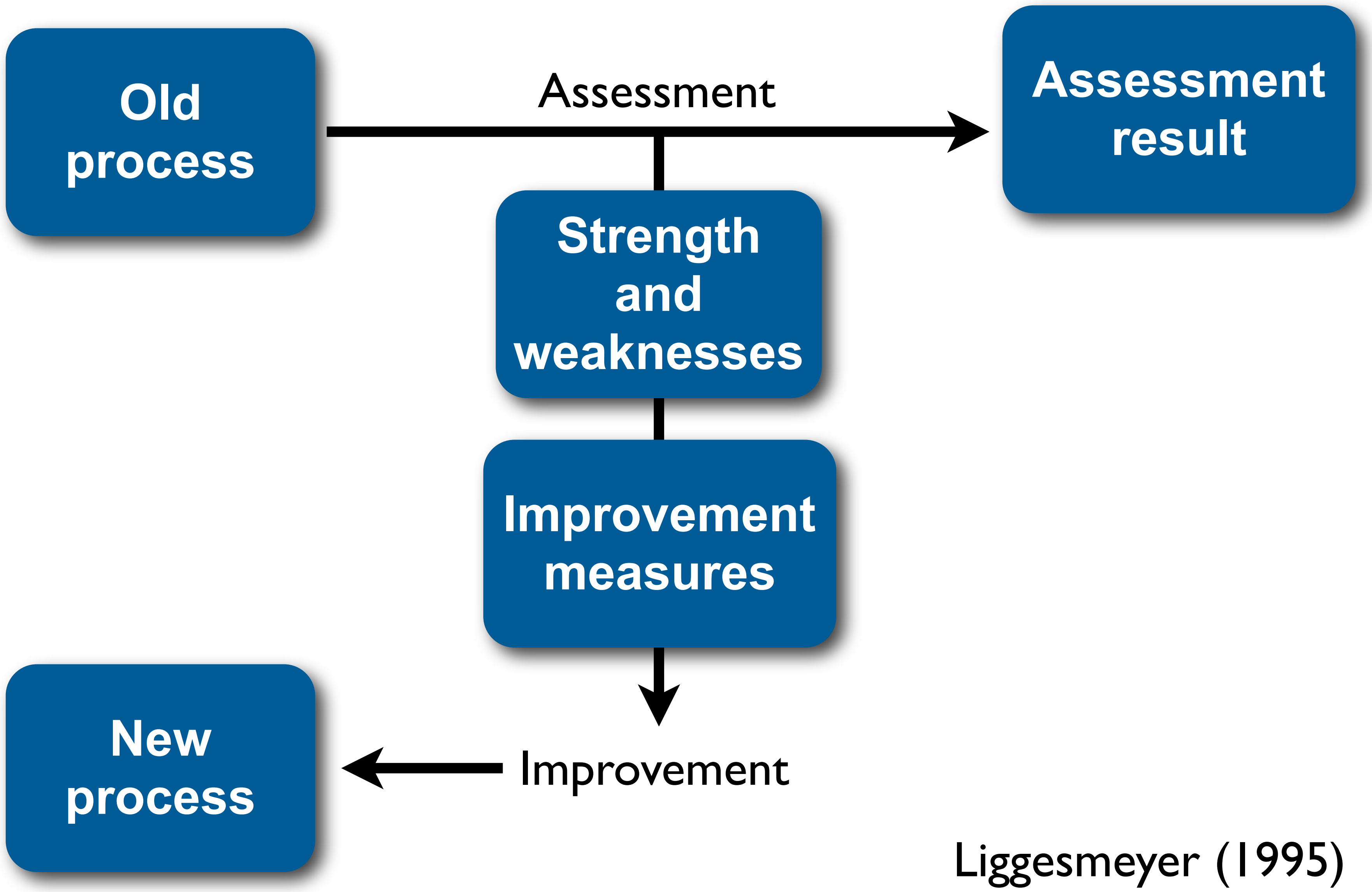
Monitoring of customer satisfaction, processes, and products.

## Post mortem analysis

ISO 9000

CMMI/SPICE

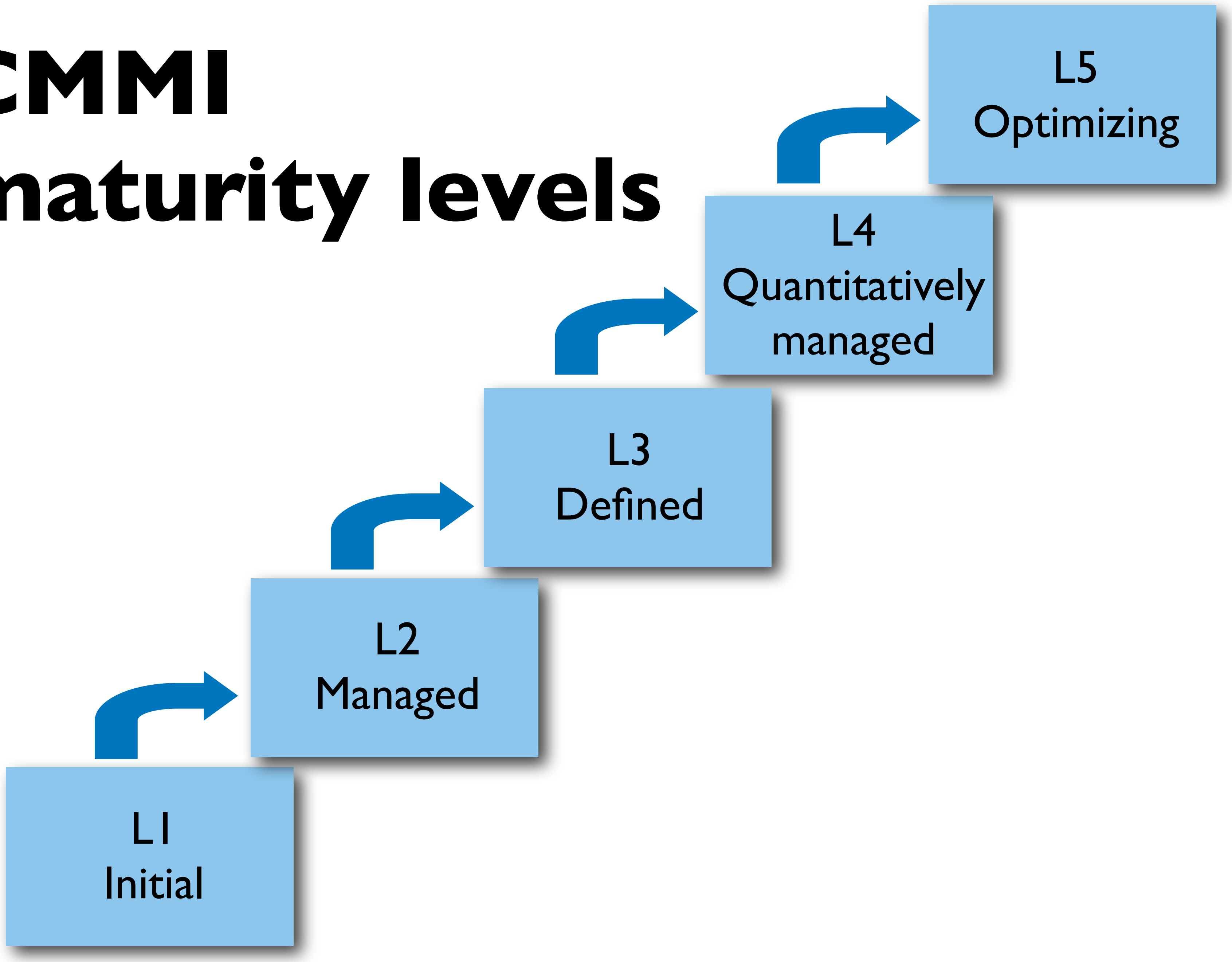
## Process improvement



Liggesmeyer (1995)

Process improvement always has the aim to move from an old process to a new process.
This is achieved by assessing the old process, which also shows strength and weaknesses of the process.
The weaknesses are the basis for deriving improvement measures.
These improvement measures are implemented, the process improves, and becomes the new process.

# CMMI maturity levels



The maturity levels in CMMI describe the level of maturity of the current process.

The initial level is ad-hoc and can be chaotic.

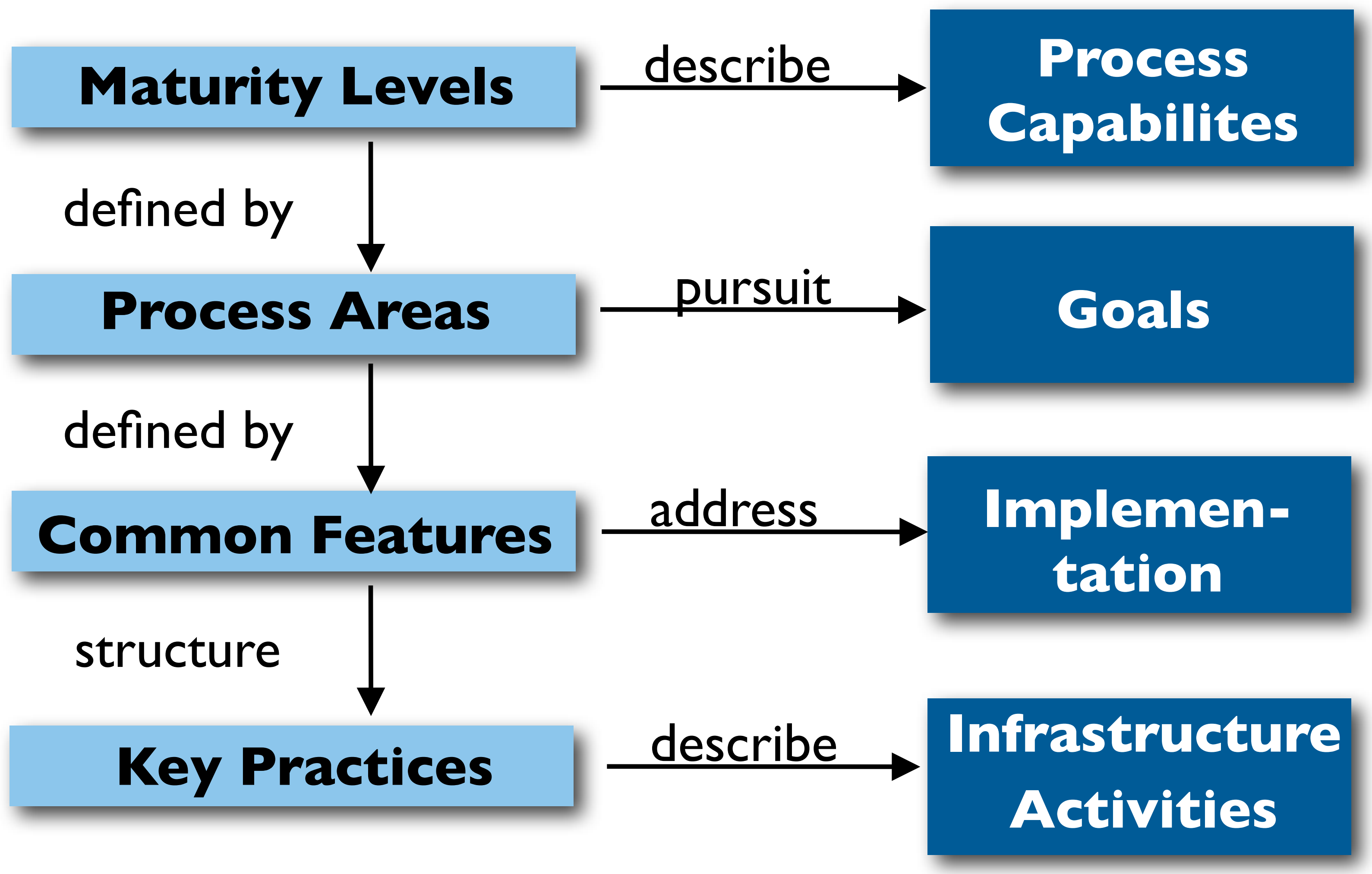
On the managed level, the actions are monitored and reacted upon the results.

On the defined level, there is a company-wide process definition.

The quantitatively managed level adds quantitative analysis to the monitoring, i.e., the managers measure and decide based on these measures.

On the optimising level, the definition of the process becomes variable again. The process is changed according to the influences of the environment so that it is optimal.

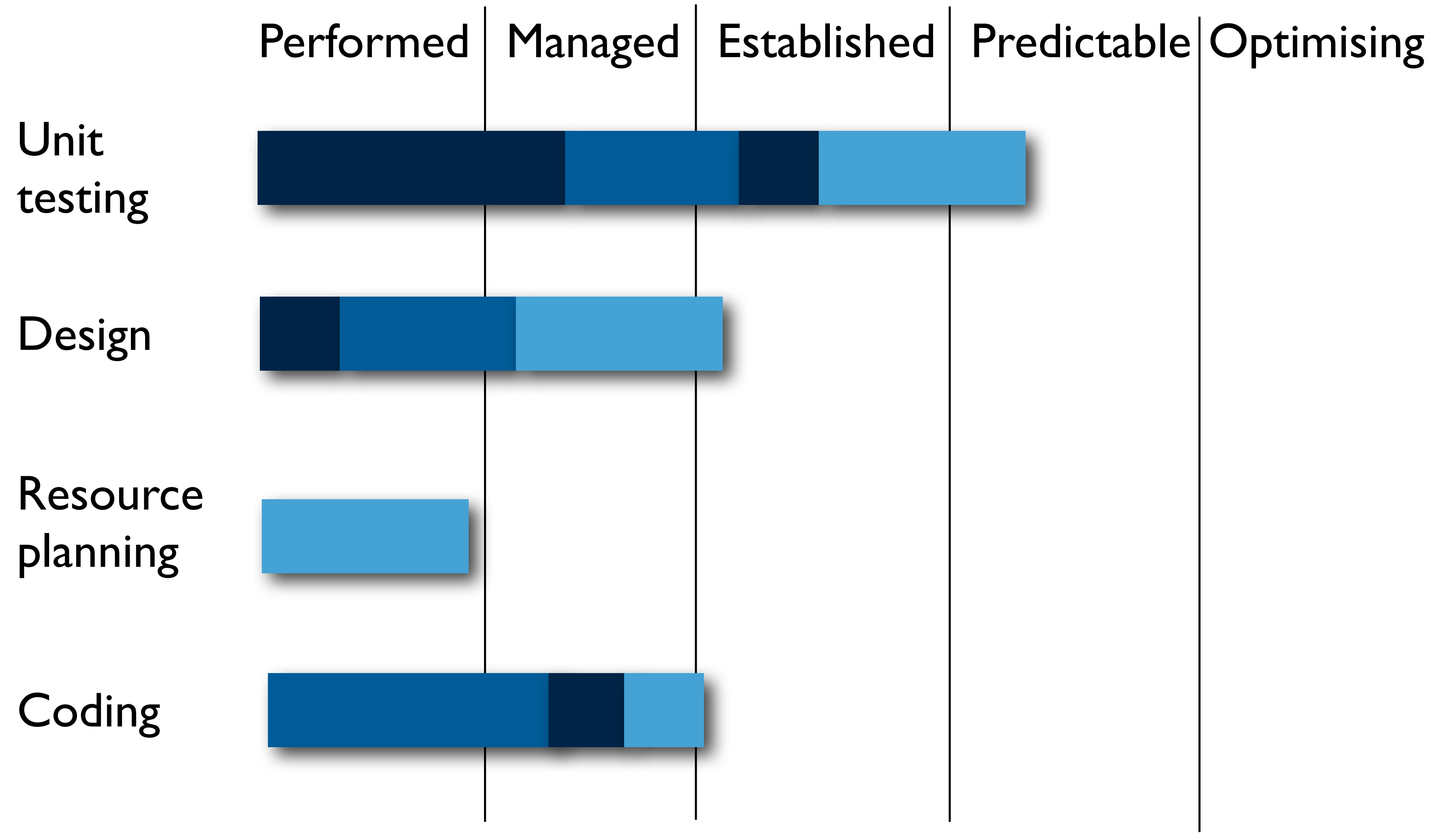
## CMMI structure



The maturity models describe process capabilities, i.e., competences in the process. These capabilities are part of process areas. The complete process is divided into these process areas. Each process area has its own goals.

Process areas can be implemented differently in different companies, but they should share common features that are structured by key practices. The key practices describe activities and infrastructure that should be part of the process.

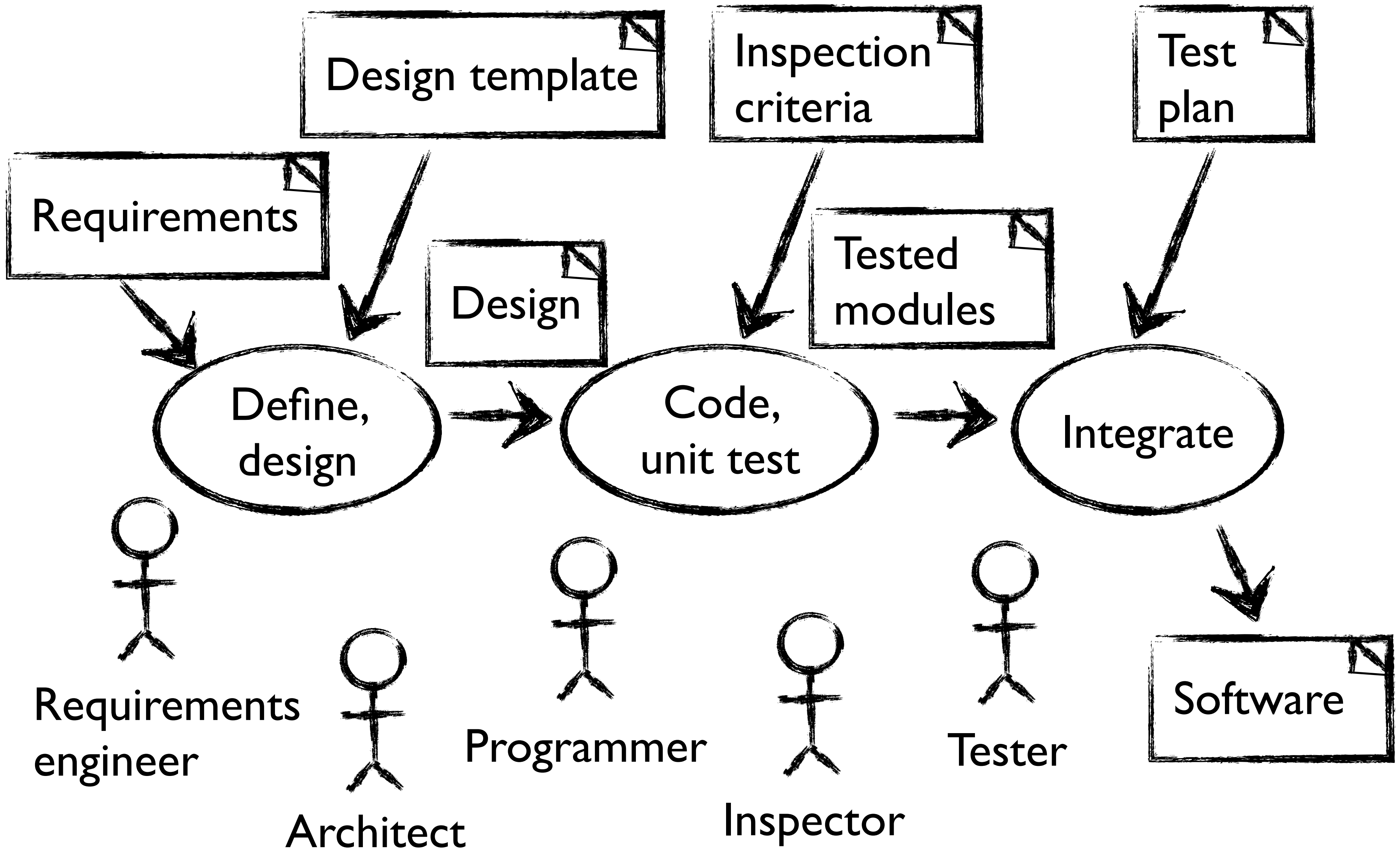
## SPICE



SPICE is the European-led ISO standard that is similar to the American CMMI. It introduced the notion that there are not only maturity levels. To reach a higher maturity level all key practices have to be implemented. The capabilities in SPICE can be achieved at different degrees:
Fully achieved
Largely achieved
Partially achieved
Not achieved
There are different maturity levels as well that need different achievements in different key practices. SPICE, however, also allows to show each capability on its own to manage process improvement in more detail.

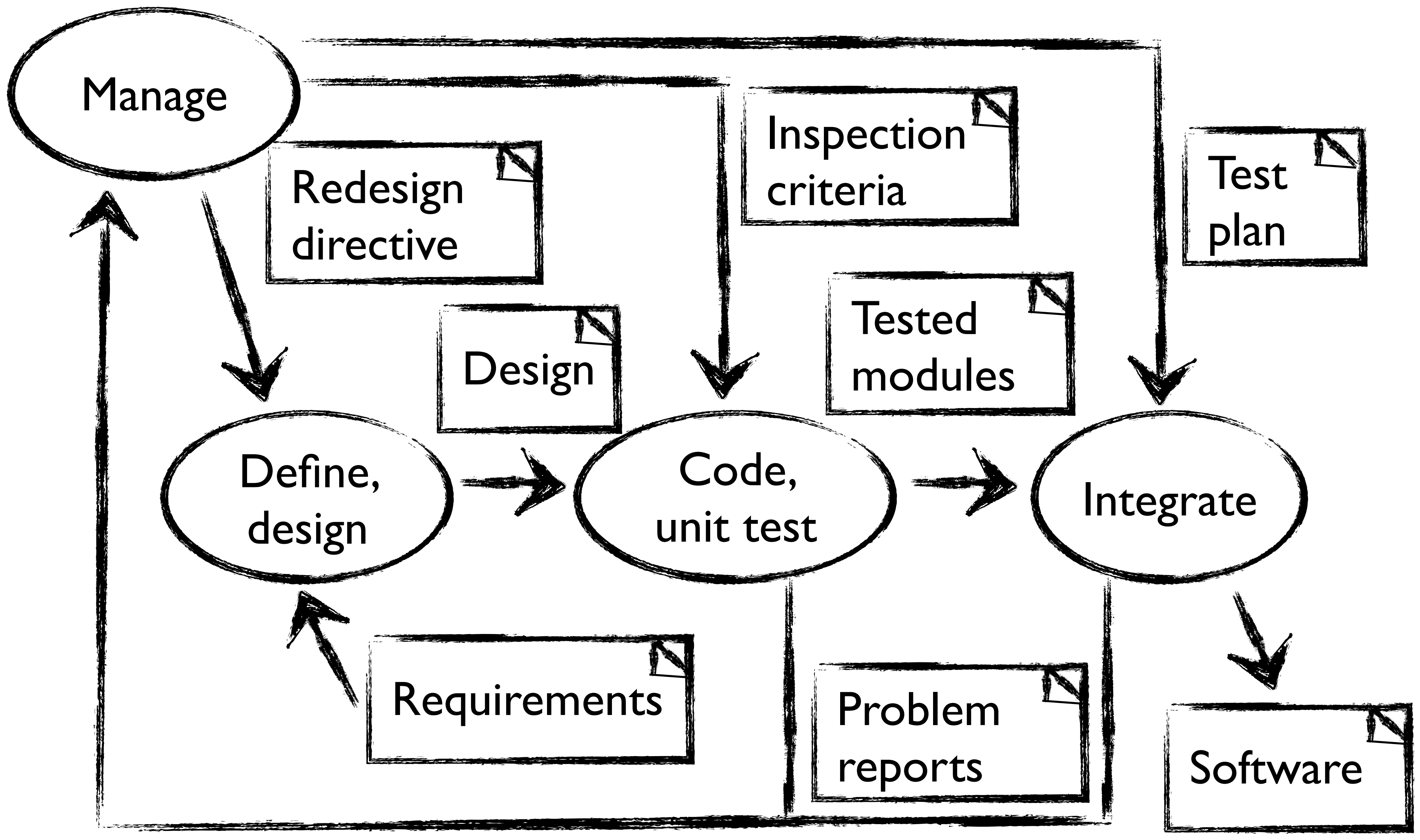
CMMI allows this by now also in its continuous representation.

## A defined process

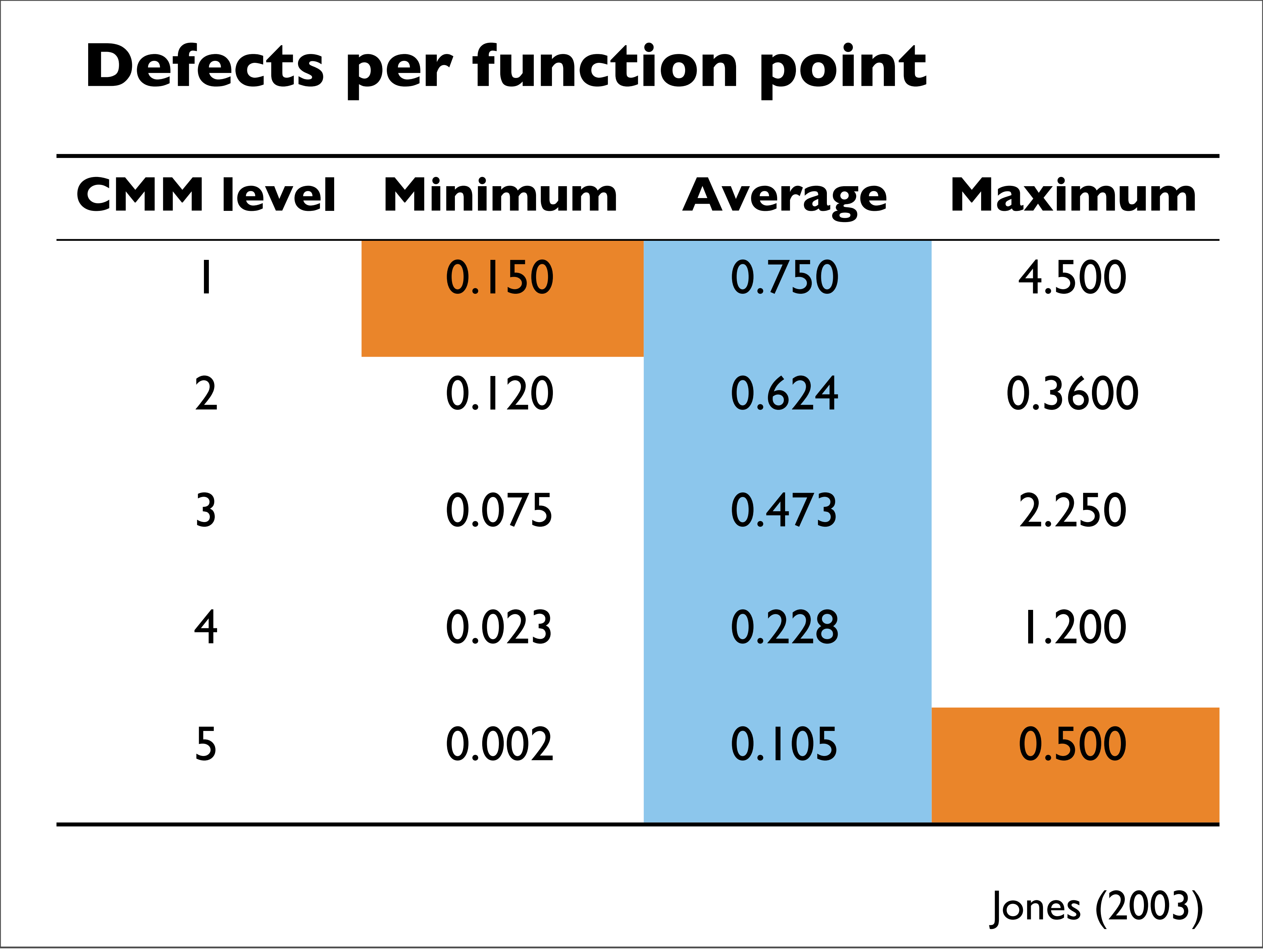


Here is the process sketch again. It is a defined process as it clearly defines the steps that have to be done.

## A managed process



The defined process can be improved to a managed process by explicitly adding a "manage" activity that receives the problem reports from each step and feeds in change directives.
This is also adapted from Pfleeger, Atlee (2010).



## Defects per function point

| CMM level | Minimum | Average | Maximum |
|---|---|---|---|
| 1 | 0.150 | 0.750 | 4.500 |
| 2 | 0.120 | 0.624 | 0.3600 |
| 3 | 0.075 | 0.473 | 2.250 |
| 4 | 0.023 | 0.228 | 1.200 |
| 5 | 0.002 | 0.105 | 0.500 |

Jones (2003)

Does CMMI actually improve processes? This has not been analysed empirical enough so that we can make conclusive statements.
Jones published an interesting analysis that shows that the defect density decreases on average if CMMI levels increase. The best CMMI level 1 companies, however, have a lower defect density than the worst CMMI level 5 companies.

## Post mortem analysis

ISO 9000

CMMI/SPICE